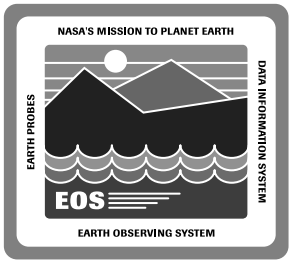
194-430-TPW-001



SDP Toolkit Implementation With Pathfinder SSM/I Precipitation Rate Algorithm

Technical Paper

Technical Paper—Not intended for formal review or government approval.

November 1994

Prepared Under Contract NAS5-60000

RESPONSIBLE ENGINEER

Narayan Prasad /s/ 11/3/94

Narayan S. Prasad, PDPS Scientist/Engineer Date
EOSDIS Core System Project

SUBMITTED BY

Parag Ambardekar /s/ 11/3/94

Parag Ambardekar, PDPS Manager Date
EOSDIS Core System Project

Hughes Applied Information Systems
Landover, Maryland

Name: Parag Ambardekar

Planning & Data Processing System (PDPS) Manager

Hughes Applied Information Systems, Inc.

Landover, MD 20785

As part of the prototyping activities at the ECS Science and Technology Lab (STL), we plan to acquire, port and run algorithm code from various instruments on the distributed/parallel testbed and HPCC sites to the extent possible. The purpose of this activity is to assess various processing technologies (DCE, workstation multiprocessors, MPPs, etc.), evaluate and validate hardware architecture for PGS, and share lessons learned with the science community. It is also intended to identify issues for resolution to facilitate algorithm integration and testing. We plan to communicate the progress on these activities and important lessons learned for hardware architecture and algorithm integration and testing from time to time through informal reports.

The reports are meant for informal exchange of technical information rather than formal ECS programmatic status. Consequently the plans for future activities may change based on schedule and resource constraints. These reports represent a snapshot of PGS activities in ECS STL at a given time. We do not spend time and resources to polish the reports and present the information in a logical manner. The attached report is entitled "SDP Toolkit Implementation With Pathfinder SSM/I Precipitation Rate Algorithm".

Note: PDPS was formerly Product Generation System (PGS)

SDP Toolkit Implementation with Pathfinder SSM/I Precipitation Rate Algorithm

Narayan Prasad

Marek Chmielowski

(November 1, 1994)

1. Background

The SDP Toolkit is used by the EOS data production software developers and scientists to encapsulate their science software for implementation in the Distributed Active Archive Center (DAAC) computing facilities. It provides an interface between instrument data processing software and the production system environment. Besides the SDP Toolkit facilitating smooth transition and integration of code into the DAAC by abstracting out science software dependencies on external system architectures, it must provide superior performance with low overhead, and should not degrade the performance of the science algorithm.

2. Objective

This study is aimed at studying the performance of the SDP Toolkit with the Pathfinder SSM/I precipitation rate algorithm (Fortran 77) obtained from NASA/MSFC[1]. The lessons learned during implementation of the toolkit with the algorithm are outlined. The ability to use toolkit functions in a parallel symmetric multiprocessing (SMP) environment is also investigated. The error handling functions in SDP Toolkit 3 do not currently support algorithm execution in a concurrent processing paradigm such as SMP, Massively Parallel Processing (MPP), Distributed Computing Environment (DCE), etc. This strategy is aimed at conserving resources that may be needlessly spent should no instrument team decide to use SMP, MPP, DCE, etc. We will consider adding such capability to the SDP Toolkit should it be necessary. This additional capability will be at lower levels in the Toolkit and should not affect the user API. This report outlines strategies (using compiler directives) that can be applied for using the toolkit even in a concurrent environment using symmetric multiprocessors. Only a subset of the SDP Toolkit functions that are relevant to the Pathfinder SSM/I precipitation rate algorithm are used. Nonetheless, this study validates the use of the SDP Toolkit for science processing.

3. Hardware

An 8-processor SGI Challenge XL (150 MHz) was used to evaluate the performance of the algorithm with the SDP Toolkit. The CHALLENGE XL is the high-end member of the CHALLENGE line. The XL server supports coherent shared memory and symmetric multiprocessing based on 100 MHz and 150 MHz MIPS RISC R4400MC 64-bit microprocessors. The server can be configured with 2 to 36 CPUs, 64 MB to 16 GB of main memory, 2 GB to 3.4 TB of disk capacity, one to four 320 MB/sec I/O channels, and 5 to 25 industry-standard VME64 bus slots.

4. Performance Study

The SGI performance profiler was used for the analysis. Table 1 outlines the performance of user coded toolkit functions. Inclusive time means, time spent within the calling function including all calls made from it. Exclusive times are not given because the timer resolution did not give significant digits to quantify measurements accurately. Only toolkit functions relevant to SSM/I processing have been used. Geolocation, coordinate transformation, and time and date functions

were not evaluated. Approximately 33% of the functions as part of the SDP Toolkit have been used in the Pathfinder SSM/I precipitation rate algorithm. This algorithm also uses 50% of all mandatory functions in the SDP Toolkit.

Table 2 lists the performance of toolkit functions that were implicitly tested. They are called by explicit user coded functions listed in Table 1.

Table 1: Performance of explicitly called toolkit functions

Toolkit Function	Mandatory(M) or Optional(O)	Number of Function calls	CPU time in seconds (inclusive)
PGS_PC_GetReference()	M	3	0.966
PGS_SMF_SetStaticMessage()	M	122	0.544
PGS_PC_GetNumberOfFiles()	O	1	0.058
PGS_SMF_SetDynamicMessage()	O	Used only for error handling	
PGS_SMF_GetMsg()	O	1	0.006
PGS_SMF_GetMsgByCode()	O	Used only for error handling	
PGS_SMF_TestStatusLevel()	O	Used only for error handling	
PGS_SMF_TestSuccessLevel()	O	Used only for error handling	
PGS_SMF_TestErrorLevel()	O	5	0.000
PGS_SMF_TestFatalLevel()	O	2	0.000
PGS_SMF_Test_NoticeLevel()	O	6	0.000
PGS_SMF_Test_UserInfoLevel()	O	6	0.000
PGS_SMF_SendStatusReport()	O	1	Timing is dependent on network activity
PGS_SMF_SendRunTimeData()	O	1	Timing is dependent on network activity

Table 2: Performance of implicitly called toolkit functions

Toolkit Function	Number of Function calls	CPU time in seconds (inclusive)
PGS_SMF_GetGlobalVar()	261	0.771
PGS_IO_GenOpen()	3	0.659
PGS_PC_GetPCSData	18	0.966
PGS_PC_GetDataAdvancedArea()	18	0.302
PGS_PC_GetPCSDataGetFileName()	6	0.080
PGS_PC_GetPCSDataIndex()	16	0.126
PGS_PC_GetPCSDataGetRequest()	13	0.058
PGS_PC_GetPCSDataLocalEntry()	18	0.508
PGS_PC_GetPCSDataOpenPCSFile()	18	0.092
PGS_PC_GetDataRetrieveData()	8	0.123
PGS_SMF_CreateMsgTag()	1	0.112
PGS_SMF_DecodeCode()	33	0.439
PGS_SMF_ExtractFileInfo()	33	0.006
PGS_SMF_ExtractMsgInfo()	107	0.023
PGS_SMF_GetEnv()	33	0.000
PGS_SMF_GetGlobalVar()	261	0.771
PGS_SMF_WriteLogFile()	33	0.102

4.1 Overall performance

The algorithm embedded with relevant toolkit calls was run in serial mode. Table 3 lists the performance characteristics. The toolkit did not introduce any significant overhead. Over 90% CPU utilization indicates that the processor was dedicated to this run.

The algorithm was also run on 8 processors and timed. Using compiler directives, portions of the algorithm which contained toolkit calls were forced to run only in serial mode (on a single processor), because the SMF library is not designed to run in parallel. The timing statistics are shown in Table 4.

At least for the Pathfinder SSM/I algorithm, the serial version of the toolkit library should suffice in a parallel environment without any major degradation in performance. When the parallel version of the algorithm was embedded with toolkit calls that use e-mail services, the performance can be

very much dependent on network traffic. Toolkit functions that use e-mail are used to send notifications of file status, etc. back to the SCFs. Table 5 indicates that with e-mail services, the algorithm can take ~6 times longer to complete. The CPU utilization drops because the parallel process is executed quickly on 8 processors, but the process of sending e-mail uses only one processor while the remaining processors are idle most of the time. We note that current toolkit e-mail services are directed at SCF development. Their implementation in a production environment may be quite different. Also, the performance results given here may not be relevent to DAAC performance.

5. Power Challenge Systems and Implications for SDP Toolkit

The fundamental difference between the CHALLENGE and Power CHALLENGE systems is that the Power CHALLENGE Server's 64-bit implementation of the IRIX Operating system supports 64-bit floating point, integer, and addressing capabilities, allowing applications to take full advantage of the hardware. This environment has implications to the SDP Toolkit which has been developed for SGI 32-bit architectures.

6. Summary

- The relevant functions from the SDP Toolkit were successfully implemented with Pathfinder SSM/I precipitation rate algorithm (Fortran 77) without introducing any noticeable overhead (geolocation, coordinate transformation, and time and date functions were not implemented). The output products were also verified.

Table 3: Performance characteristics in serial mode

Toolkit	Wall clock time (s)	User time (s)	System time (s)	CPU utilization (%)
No	231	217.5	8.6	97
Yes	243	218.0	9.1	93

Table 4: Performance characteristics for parallel mode

Toolkit	Wall Clock time (s)	User time (s)	System time (s)	CPU utilization (%)
No	34	183.1	22.6	607
Yes	34	189.4	23.0	601

Table 5: Performance with toolkit functions using e-mail services

Toolkit	Wall Clock time (s)	User time (s)	System time (s)	CPU utilization (%)
Yes	203	199.9	62.3	128

- The serial version of the toolkit can be used in a parallel environment. However, portions of the code that contain toolkit calls must be forced to run only on a single processor by using parallel compiler directives. At least with the SSM/I algorithm, when the toolkit-embedded algorithm was run in parallel, there was no noticeable degradation in performance. This is one alternative for using existing toolkit functions in a parallel environment.
- Approximately 33% of the functions as part of the SDP Toolkit have been used in the Pathfinder SSM/I precipitation rate algorithm. This algorithm also uses 50% of all mandatory functions in the SDP Toolkit.
- When SDP Toolkit functions utilizing e-mail services are used, the performance of the algorithm is strongly influenced by network traffic. Severe degradation in performance could occur. We note that current toolkit e-mail services are directed at SCF development. Their performance in a production environment may be different.
- We note that some SMF functionality involves disk I/O (e.g. writing out a log file). This operation is inherently serial with current technology. Users should design their algorithms with non-parallel functions grouped such that the advantages gained by parallelizing the remaining code are not lost.

7. References

[1] PDPS Prototyping at ECS Science and Technology Laboratory Progress Report #4. September 1994, ECS Document #194-00569TPW

This page intentionally left blank.

Appendix A: Description of SDP Tools Evaluated

Table A1: Process Control Tools

Toolkit Function	Description
PGS_PC_GetReference()	This tool may be used to obtain a physical reference (file name or universal identifier) from a logical identifier
PGS_PC_GetNumberofFiles()	This tool may be used to determine the number of files that exist for a particular product group
PGS_PC_GetPCSData()	This tool may be used to obtain information concerning the availability of PGE input files, process ID's, science software ID's, runtime parameters, etc. (It is no longer a user API. Users now have higher level functions to retrieve the required types of data)

Table A2: SMF Tools

Toolkit Function	Description
PGS_SMF_GenerateStatReport()	This tool is used to write the message string into error/status logfile. This tool is generally used in conjunction with PGS_SMF_CreateMsgTag()
PGS_SMF_SetStaticMsg()	This tool will provide the means to set a user-defined error/status message in response to the outcome of some segment of processing
PGS_SMF_GetMsgByCode()	This tool will provide the means to retrieve the message string corresponding to a specific mnemonic code
PGS_SMF_GetMsg()	This tool will provide the means to retrieve previously set message from the static buffer (PGS_SMF_set..()). It should be called immediately; otherwise, the intended message will be overwritten by subsequent calls to set message routines
PGS_SMF_CreateMsgTag()	This tool may be used to generate a unique message identifier. This identifier must be attached to error/status message string to facilitate tracking of error/status messages to module of code within a product executable, for a unique production run. This tool is generally used in conjunction with PGS_SMF_GenerateStatReport()

Table A2: SMF Tools

Toolkit Function	Description
PGS_SMF_SendStatusReport()	This tool may be used to transfer error/status report logs to key monitoring authorities following a production run
PGS_SMF_SendRuntimeData()	This tool will provide a means for the user to transmit a package of run-time data to SCF in the event of an unhandled system exception. This package may contain status/error log and volatile temporary files that may be useful for discerning the problem that may have initiated the exception. This tool should only be invoked once for the given process
PGS_SMF_TestErrorLevel()	Given the mnemonic status code, this tool will return the Boolean value indicating whether or not the returned code has level 'E'
PGS_SMF_TestFatalLevel()	Given the mnemonic status code, this tool will return a Boolean value indicating whether or not the return code has level 'F'
PGS_SMF_TestSuccessLevel()	Given the mnemonic status code, this tool will return a Boolean value indicating whether or not the return code has level 'S'
PGS_SMF_TestUserInfoLevel()	Given the mnemonic status code, this tool will return a Boolean value indicating whether or not the return code has level 'U'
PGS_SMF_TestNoticeLevel()	Given the mnemonic status code, this tool will return a Boolean value indicating whether or not the return code has level 'N'
PGS_SMF_TestStatusLevel()	Given the mnemonic status code, this tool will return a defined status level constant
PGS_SMF_SetDynamicMsg()	This tool will provide the means to set a user-defined error/status message in response to the outcome of some segment of processing. The user can also attach a message string to the defined mnemonic code, thus overriding the defined message string that was created by the smfcompile